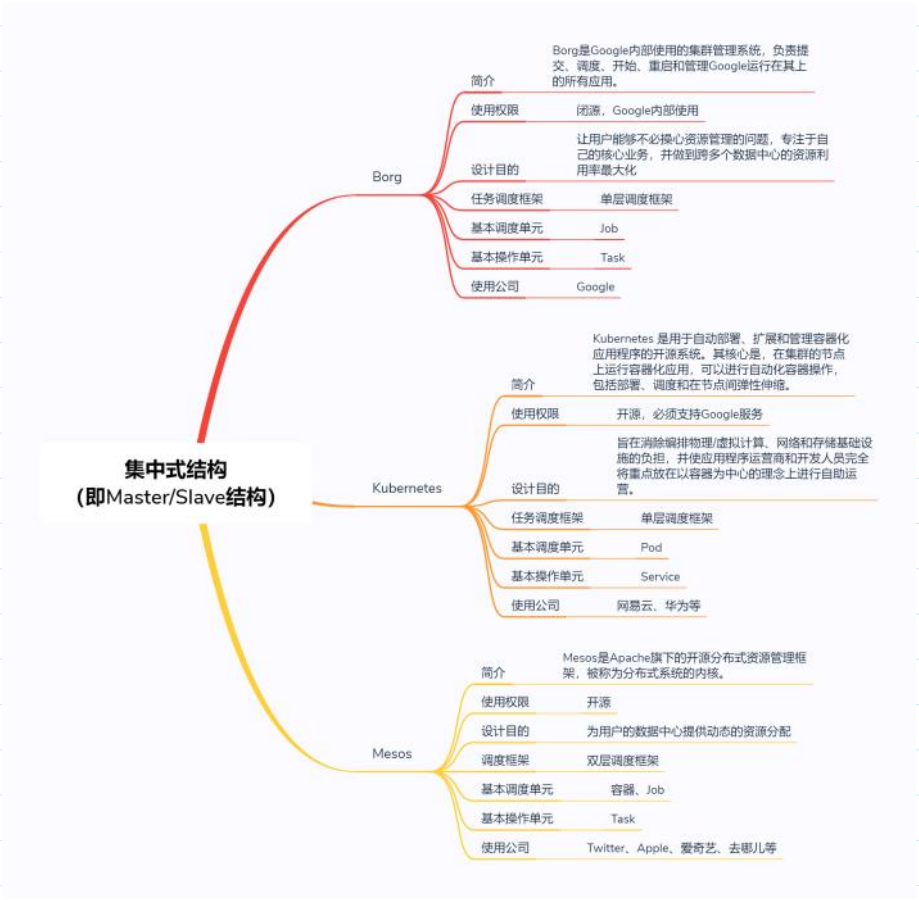
2.1集中式结构

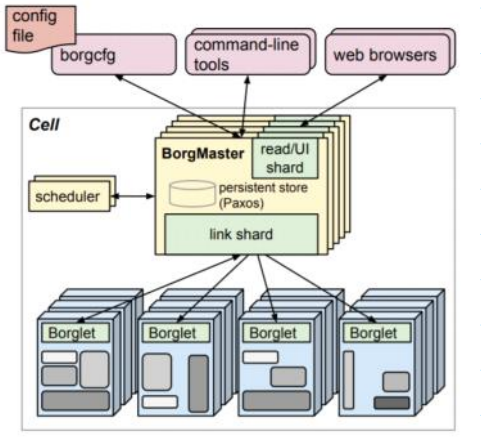
2022年2月25日 10:12



Google Borg.

①结构.

one cluster = one cell { one Borg Master. / many Borglet.

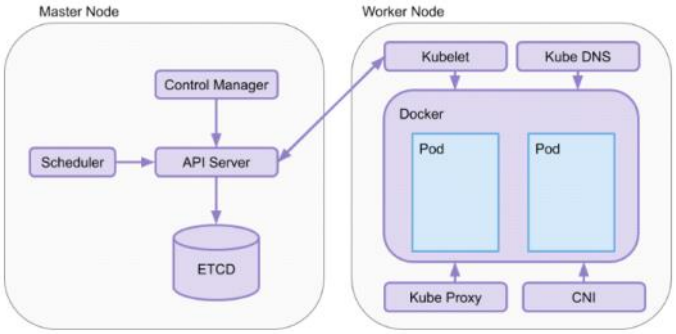


②优点.

- 开发者只需关注应用，不需要关注底层资源管理。它隐藏了资源管理以及错误处理，因此用户能集中精力开发应用。
- 高可靠性和可用性，支持多种应用。
- 支持上千级服务器的管理和运行。

kubernets.

①结构.



我们先来看看 Master 节点。它运行在中心服务器上，Master 节点由 API Server、Scheduler、Cluster State Store 和 Control Manger Server 组成，负责对集群进行调度管理。

- API Server：是所有 REST 命令的入口，负责处理 REST 的操作，确保它们生效，并执行相关业务逻辑。
- Scheduler：根据容器需要的资源以及当前 Worker 节点所在节点服务器的资源信息，自动为容器选择合适的节点服务器。
- Cluster State Store：集群状态存储，默认采用 etcd，etcd 是一个分布式 key-value 存储，主要用来做共享配置和服务发现。
- Control Manager：负责整个集群的编排管理。它监视集群中节点的离开和加入，将集群的当前状态与 etcd 中存储的所需状态进行核对。比方说，当某个节点发生故障，它会在其它节点上增加新的 Pod 以匹配所需的副本数。

接下来，我们看看 Worker 节点吧。它作为真正的工作节点，运行在从节点服务器，包括 kubelet 和 kube-proxy 核心组件，负责运行业务应用的容器。

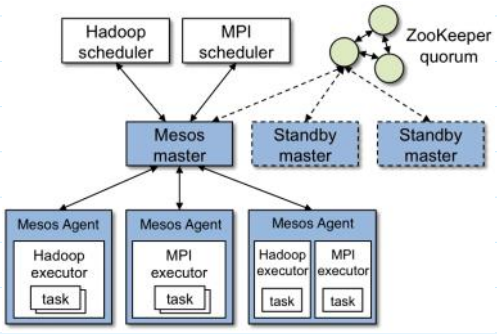
- kubelet：用于通过命令行与 API Server 进行交互，根据接收到的请求对 Worker 节点进行操作。也就是说，通过与 API Server 进行通信，接收 Master 节点根据调度策略发出的请求或命令，在 Worker 节点上管控容器（Pod），并管控容器的运行状态（比如，重新启动出现故障的 Pod）等。Pod 是 Kubernetes 的最小工作单元，每个 Pod 包含一个或多个容器。
- kube-proxy：负责为容器（Pod）创建网络代理 / 负载平衡服务，从 API Server 获取所有 Server 信息，并根据 Server 信息创建代理服务，这种代理服务称之为 Service。Kube-proxy 主要负责管理 Service 的访问入口，即实现集群内的 Pod 客户端访问 Service，或者是集群外访问 Service，具有相同服务的一组 Pod 可抽象为一个 Service。每个 Service 都有一个虚拟 IP 地址（VIP）和端口号供客户端访问。

②优点.

- **自动化容器的部署和复制。**Kubernetes 执行容器编排，因此不必人工编写这些任务的脚本。
- **将容器组织为组，弹性伸缩。**Kubernetes 引入 Pod 机制，Pod 代表着能够作为单一应用程序加以控制的一组容器集合。通过 Pod 机制，Kubernetes 实现了多个容器的协作，能够有效避免将太多功能集中到单一容器镜像这样的错误实践中。此外，软件可以向外扩展跨越多个 Pods 实现初步部署，且相关部署可随时进行规模伸缩。
- **容器间负载均衡。**Services 用于将具备类似功能的多个 Pod 整合为一组，可轻松进行配置以实现其可发现性、可观察性、横向扩展以及负载均衡。
- **易于版本控制与滚动更新。**Kubernetes 采取“滚动方式”实现编排，且可跨越部署范围内的全部 Pod。这些滚动更新可进行编排，并以预定义方式配合当前可能尚不可用的 Pods 数量，以及暂时存在的闲置 Pods 数量。Kubernetes 利用新的应用程序镜像版本对已部署 Pods 进行更新，并在发现当前版本存在不稳定问题时回滚至早期部署版本。

Mesos

①结构.



②优点.

- **效率。**Mesos 对物理资源进行了逻辑抽象，在应用层而不是物理层分配资源，通过容器而不是虚拟机（VM）分配任务。因为应用程序的调度器知道如何最有效地利用资源，所

- **效率。**Mesos 对物理资源进行了逻辑抽象，在应用层而不是物理层分配资源，通过容器而不是虚拟机（VM）分配任务。因为应用程序的调度器知道如何最有效地利用资源，所以在应用层分配资源能够为每个应用程序的特殊需求做考量；而通过容器分配任务则能更好地进行“装箱”。
- **可扩展性。**Mesos 可扩展设计的关键是两级调度架构，其中 Framework 进行任务调度，Mesos Master 进行资源分配。由于 Master 不必知道每种类型的应用程序背后复杂的调度逻辑，不必为每个任务做调度，因此可以用非常轻量级的代码实现，更易于扩展集群规模。
- **模块化。**每接入一种新的框架，Master 无需增加新的代码，并且 Agent 模块可以复用，为此开发者可以专注于应用和框架的选择。这，就使得 Mesos 可以支持多种框架，适应不同的应用场景。